Software Requirements Specification

for

InvIQ- Inventory Management system

Version 1.0 approved

Prepared by
Nisarg Chaudhari
Probin Bhagchandani
Urva Dave
Japan Kachhiya

CHARUSAT University

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The INVIQ Inventory Management System is designed to revolutionize inventory operations for restaurant owners and managers, aiming to streamline processes, monitor sales transactions, optimize inventory levels, and maximize profitability. By providing intuitive features such as simplified inventory management, real-time sales monitoring, predictive selling capabilities, and operational efficiency enhancements, the system empowers users to make data-driven decisions and improve overall performance. With advanced technology and predictive analytics at its core, INVIQ aims to address the challenges faced by restaurants in managing inventory, reducing wastage, and enhancing customer satisfaction, ultimately driving success in the competitive food service industry.

1.2 Intended Audience and Reading Suggestions

The Software Requirement Specification (SRS) document for the INVIQ Inventory Management System is tailored for a wide audience involved in various aspects of the system's development, implementation, and usage. This includes stakeholders such as restaurant owners, managers, investors, developers, quality assurance professionals, project managers, and end users like restaurant staff. Stakeholders can refer to this document to gain a comprehensive understanding of the system's functionalities, requirements, and benefits. Developers can utilize it as a blueprint for designing and implementing the system, while quality assurance professionals can use it to define testing criteria and ensure the system's reliability. Project managers can leverage the document to manage resources effectively and monitor project progress, while end users can refer to it to understand how the system will enhance their day-to-day operations.

1.3 Product Scope

The product scope of the INVIQ Inventory Management System encompasses a comprehensive set of features and functionalities aimed at optimizing inventory operations within the restaurant industry. The system enables users to efficiently manage inventory items, track stock levels, monitor sales transactions, and make informed decisions based on predictive analytics. Key components of the product scope include inventory management features such as adding, updating, and deleting inventory items, as well as setting alerts for low stock levels. Additionally, the system provides sales monitoring capabilities, allowing users to track sales transactions in real-time and generate reports on sales performance. Furthermore, the product scope extends to predictive selling features, utilizing machine learning algorithms to analyze historical sales data and forecast future sales trends. Overall, the INVIQ Inventory Management System offers a robust solution to streamline inventory operations, reduce wastage, and maximize profitability for restaurants.

1.4 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2. Overall Description

2.1 Product Perspective

The INVIQ Inventory Management System operates as an integral component within the broader ecosystem of restaurant management systems. It interfaces seamlessly with existing point-of-sale (POS) systems, accounting software, and supplier databases to facilitate the smooth flow of data and transactions. As a standalone application, the system provides a user-friendly interface for restaurant owners, managers, and staff to efficiently manage inventory, monitor sales transactions, and make data-driven decisions. It complements other restaurant management tools by enhancing inventory control, reducing wastage, and optimizing stock levels. The system's predictive selling capabilities offer a forward-looking perspective by leveraging historical sales data and machine learning algorithms to forecast future demand, thereby enabling proactive inventory management. In essence, the INVIQ Inventory Management System plays a pivotal role in the operational efficiency and profitability of restaurants within the broader context of the hospitality industry.

2.2 Product Functions

2.2 Product Functions:

The INVIQ Inventory Management System offers a comprehensive range of functionalities tailored to streamline inventory operations and optimize efficiency within the restaurant environment. Key product functions include:

1. Inventory Management: Enable users to add, edit, and delete inventory items, as well as categorize them for efficient organization. Track stock levels in real-time and set alerts for low inventory thresholds to ensure timely replenishment.

2. Sales Monitoring: Provide a dashboard for real-time monitoring of sales transactions, including order details, customer information, and payment status. Generate comprehensive reports on sales performance to facilitate data-driven decision-making.

3. Predictive Selling: Utilize machine learning algorithms to analyze historical sales data and predict future demand trends. Provide insights into anticipated sales volumes to enable proactive inventory management and reduce stockouts or overstocking.

4. Alerts and Notifications: Send automated alerts and notifications to users for critical inventory events, such as low stock levels, pending orders, and predictive sales alerts. Keep stakeholders informed and enable timely responses to inventory-related issues.

5. Reporting and Analytics: Generate customizable reports and analytics dashboards to analyze sales trends, inventory turnover rates, and profitability metrics. Provide insights into key performance indicators to identify areas for improvement and optimize business processes.

6. Integration Capabilities: Seamlessly integrate with existing point-of-sale (POS) systems, accounting software, and supplier databases to facilitate data exchange and streamline workflow processes. Ensure compatibility with industry-standard formats for interoperability and ease of integration.

7. User Management: Administer user roles and permissions to control access to system functionalities based on organizational roles and responsibilities. Ensure data security and privacy by implementing robust authentication and authorization mechanisms.

.

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

1. Hardware Requirements: The system is compatible with standard computing devices such as desktop computers, laptops, tablets, and mobile devices. It requires sufficient processing power and memory to support the application's functionalities and ensure smooth performance.

2. Operating System: The system is platform-independent and can run on various operating systems, including Windows, macOS, Linux, and mobile operating systems such as iOS and Android.

3. Web Browser Compatibility: The system is accessed through a web-based interface, making it compatible with popular web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. The interface is designed using responsive web design principles to ensure optimal viewing and functionality across different screen sizes and resolutions.

4. Internet Connection: A stable and reliable internet connection is essential for accessing the system and synchronizing data in real-time. While offline functionality may be available for certain features, uninterrupted internet connectivity is recommended for optimal performance.

5. Database Management System (DBMS): The system relies on a relational database management system (RDBMS) for storing and managing data related to inventory items, sales transactions, user accounts, and system configurations. Compatible DBMS options include MySQL, PostgreSQL, SQLite, or Microsoft SQL Server.

6. Security Measures: The operating environment should adhere to industry-standard security protocols and best practices to ensure the confidentiality, integrity, and availability of data. This includes implementing encryption mechanisms, access controls, firewalls, and regular security updates to mitigate potential security risks.

2.5 Design and Implementation Constraints

1. Technology Stack: The system must be developed using technologies and frameworks that are compatible with the target operating environment and meet the performance requirements of the application. Constraints related to technology choices, such as programming languages, databases, and third-party libraries, may impact development timelines and resource availability.

2. Budgetary Constraints: Development costs, including expenses related to software licenses, hardware infrastructure, and personnel, must be managed within budgetary constraints. Limitations on financial resources may influence decisions regarding feature prioritization, development methodologies, and resource allocation.

3. Time Constraints: The project must adhere to predefined timelines and deadlines for development, testing, and deployment phases. Time constraints may impact the scope of the project, requiring prioritization of essential features and functionalities to meet project milestones and deliverables.

4. Integration Requirements: The system must integrate seamlessly with existing restaurant management systems, POS systems, and external APIs to facilitate data exchange and interoperability. Constraints related to compatibility, data formats, and communication protocols may necessitate additional development efforts and coordination with external stakeholders.

5. Scalability and Performance: The system must be designed to accommodate growth in the number of users, transactions, and data volume over time. Constraints related to scalability and performance may require optimization of database queries, caching mechanisms, and system architecture to ensure responsiveness and reliability under increasing loads.

6. Regulatory Compliance: The system must adhere to industry-specific regulations and compliance standards, such as data privacy laws, food safety regulations, and accounting practices. Constraints related to regulatory compliance may necessitate additional features, security measures, and documentation to ensure legal and regulatory obligations are met.

7. User Experience Constraints: The system must provide an intuitive and user-friendly interface for restaurant staff and management to perform tasks efficiently. Constraints related to usability and accessibility may require iterative design and testing processes to address user feedback and ensure a positive user experience.

2.6 User Documentation

1. User Manuals: Detailed user manuals will provide step-by-step instructions for navigating the system, performing common tasks, and utilizing advanced features. Manuals will be available in both digital and printable formats for ease of access and reference.

2. Online Help: Interactive online help resources will be integrated within the system interface to provide contextual assistance and guidance to users as they navigate through different screens and functionalities. Online help will feature searchable topics, tooltips, and hyperlinks to relevant sections for quick access to information.

3. Tutorials: Video tutorials and walkthroughs will be created to demonstrate key features and workflows within the system. Tutorials will cater to users with varying levels of expertise, providing introductory overviews for beginners and in-depth demonstrations for advanced users.

4. FAQs (Frequently Asked Questions): A curated list of frequently asked questions and troubleshooting tips will be compiled to address common user queries and issues. FAQs will be accessible through the system interface and the official support portal.

5. Online Knowledge Base: An online knowledge base will be established to serve as a centralized repository of documentation, articles, best practices, and release notes. The knowledge base will be continuously updated to reflect changes in the software and address emerging user needs.

7. *Release Notes: Detailed release notes will accompany software updates and new feature releases, highlighting changes, enhancements, and bug fixes. Release notes will provide transparency and insight into the evolution of the software over time*

2.7 Assumptions and Dependencies

Assumptions:

1. Third-Party Integrations: It is assumed that third-party APIs or components required for integrating with existing restaurant management systems, POS systems, and supplier databases will be available and compatible with the system's requirements. Any changes or limitations in these integrations could impact the functionality and performance of the INVIQ Inventory Management System.

2. Data Accuracy: The accuracy and reliability of historical sales data used for predictive analytics are assumed to be consistent and trustworthy. Any discrepancies or inaccuracies in the data could affect the accuracy of sales forecasts and predictive insights generated by the system.

3. Operating Environment Stability: The stability and reliability of the operating environment, including hardware infrastructure, operating systems, web browsers, and network connectivity, are assumed to remain consistent throughout the system's lifecycle. Changes or disruptions in the operating environment could impact system performance and availability

4. Regulatory Compliance: It is assumed that the system will comply with relevant industry standards, regulations, and best practices, such as data privacy laws and food safety regulations. Any changes or updates to regulatory requirements could necessitate modifications to the system's features and functionalities.

Dependencies:

1. Third-Party Libraries and Frameworks: The project depends on the availability and compatibility of third-party libraries and frameworks, such as machine learning libraries for predictive analytics and web development frameworks for the system's frontend and backend components.

2. External APIs: The project relies on the availability and stability of external APIs for integrating with external systems and services, such as payment gateways, supplier databases, and geographic information systems (GIS) for location-based data.

3. Database Management System (DBMS): The project depends on the selection and implementation of a suitable database management system (DBMS) for storing and managing data related to inventory items, sales transactions, and user accounts. Any changes or limitations in the chosen DBMS could impact system functionality and performance.

4. Development Team Resources: The project's timeline and deliverables depend on the availability and expertise of the development team members responsible for designing, coding, testing, and deploying the INVIQ Inventory Management System. Any changes in team composition or resource availability could affect project timelines and milestones.

3. External Interface Requirements

3.1 User Interfaces

1. Dashboard: Upon logging in, users will be greeted with a customizable dashboard displaying real-time metrics, such as current inventory levels, sales trends, and alerts for low stock items. The dashboard will provide at-a-glance insights into key performance indicators and actionable information for decision-making.

2. Inventory Management Interface: This interface will allow users to add, edit, and delete inventory items, view detailed product information, and categorize items for easy organization. Users can also set alerts for low inventory levels and track stock adjustments, transfers, and returns.

3. Sales Monitoring Interface: Users can monitor sales transactions in real-time, view order details, customer information, and payment status. The interface will include features for generating reports on sales performance, analyzing trends, and identifying opportunities for revenue growth.

4. Predictive Selling Interface: This interface will leverage machine learning algorithms to analyze historical sales data and predict future demand trends. Users can access predictive insights and recommendations for optimizing inventory levels, pricing strategies, and menu offerings.

5. Alerts and Notifications: The system will display alerts and notifications for critical inventory events, such as low stock levels, pending orders, and predictive sales alerts. Notifications will be presented in a prominent manner to ensure timely action by users.

6. Help and Support: Standard buttons for accessing help resources, FAQs, and support contact information will be available on every screen to assist users in resolving issues or seeking additional assistance. Keyboard shortcuts may be provided for common actions to enhance user efficiency.

7. Error Message Display: Error messages will follow standard display conventions, providing clear and concise explanations of errors encountered and suggesting possible solutions. Error messages will be displayed in a consistent format and location for easy identification and resolution.

3.2 Hardware Interfaces

1. Computing Devices: The system supports a wide range of computing devices, including desktop computers, laptops, tablets, and mobile devices. Users can access the system using compatible web browsers installed on their devices, such as Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari.

2. Operating Systems: The system is platform-independent and can run on different operating systems, including Windows, macOS, Linux, iOS, and Android. Users can access the system from devices running these operating systems without any compatibility issues.

3. Input Devices: Users interact with the system using standard input devices such as keyboards, mice, touchscreens, and styluses. Input devices enable users to enter data, navigate through the system interface, and perform actions such as clicking, scrolling, and typing.

4. Barcode Scanners: The system interfaces with barcode scanners for scanning product barcodes during inventory management tasks, such as adding new items, updating quantities, and processing sales transactions. Barcode scanners communicate with the system via USB or wireless connections using industry-standard protocols.

5. Printers: The system interfaces with printers for generating reports, receipts, labels, and other printed materials. Users can configure the system to print documents directly from the interface, and the system communicates with printers using standard printer drivers and protocols.

3.3 Software Interfaces

1. Database Management System (DBMS): The system interacts with a relational database management system (RDBMS) for storing and managing data related to inventory items, sales transactions, user accounts, and system configurations. Examples of compatible DBMS platforms include MySQL, PostgreSQL, SQLite, or Microsoft SQL Server. Data items exchanged with the DBMS include product details, sales records, user authentication data, and system configurations.

2. APIs and Libraries: The system may utilize third-party APIs, libraries, and frameworks for additional functionality, such as machine learning algorithms for predictive analytics or payment gateways for processing transactions. Examples of APIs and libraries include TensorFlow for machine learning, Stripe for payment processing, and Google Maps for geolocation services. Data items exchanged with APIs and libraries include request parameters, response data, and authentication tokens.

3. Integrated Commercial Components: The system may integrate with commercial components or software as a service (SaaS) platforms for specific features or services. Examples include integration with accounting software for financial reporting, supplier databases for inventory procurement, and customer relationship management (CRM) systems for customer data management. Data items exchanged with integrated commercial components include customer information, financial transactions, and inventory procurement data.

System Features

4.1 Add Inventory Items
4.1.1 Description and Priority
This feature allows users to add new inventory items to the system. It is of High priority as it is essential for maintaining an up-to-date inventory.

4.1.2 Stimulus/Response Sequences
- User selects "Add Inventory Items" from the main menu.
- System displays a form for entering details of the new item.
- User fills out the form with item information.
- User submits the form.
- System validates the input and adds the new item to the inventory.
- System displays a confirmation message upon successful addition.

4.1.3 Functional Requirements
- REQ-1: The system presents a form for users to input details such as item name, description, category, quantity, and price.
- REQ-2: Users can submit the form to add the new item to the inventory.
- REQ-3: The system validates the input data to ensure completeness and accuracy.
- REQ-4: Upon successful addition, the system updates the inventory database with the new item.
- REQ-5: The system provides feedback to the user, confirming the successful addition of the item.

4.2 Edit Inventory Items
4.2.1 Description and Priority
This feature enables users to modify existing inventory items. It is of High priority as it allows for accurate inventory management.

4.2.2 Stimulus/Response Sequences
- User selects "Edit Inventory Items" from the main menu.
- System displays a list of existing inventory items.
- User selects an item to edit.
- System retrieves the item details and displays them in an editable form.
- User makes changes to the item details.
- User submits the form.
- System validates the input and updates the item in the inventory.
- System displays a confirmation message upon successful update.

4.2.3 Functional Requirements

- REQ-1: The system allows users to select an existing inventory item for editing.
- REQ-2: Users can modify the details of the selected item, including name, description, category, quantity, and price.
- REQ-3: The system validates the edited data to ensure consistency and accuracy.
- REQ-4: Upon successful validation, the system updates the inventory database with the edited item.
- REQ-5: The system provides feedback to the user, confirming the successful update of the

Other Nonfunctional Requirements

3.4 Performance Requirements

1. Response Time: The system should respond to user interactions, such as adding or editing inventory items, generating reports, and processing sales transactions, within 2 seconds under normal operating conditions. This ensures that users can perform tasks quickly and efficiently *without experiencing delays*

2. Scalability: The system should be capable of handling a growing volume of inventory items, sales transactions, and concurrent user sessions. It should scale horizontally and vertically to accommodate increased data load and user traffic without degradation in performance. The system should be able to handle at least 1000 concurrent users and 100,000 inventory items without significant performance impact.

3. Database Performance: Database queries and transactions should be optimized to ensure fast retrieval and manipulation of data. The system should be able to process complex queries, such as generating sales reports or analyzing inventory trends, within 5 seconds. Database indexing, query optimization, and caching mechanisms should be implemented to improve query performance and reduce latency.

4. Network Performance: The system's communication with external APIs, services, and databases should be efficient and reliable. Network latency should be minimized to ensure timely data exchange and response. The system should have an average network latency of less than 100 milliseconds for interactions with external services.

5. Concurrent User Handling: The system should support multiple concurrent user sessions without degradation in performance or response time. It should be able to handle peak loads during busy periods, such as restaurant opening hours, without experiencing slowdowns or system crashes. Load testing should be conducted to verify the system's ability to handle concurrent user loads of up to 500 users.

3.5 Software Quality Attributes

3.5 Software Quality Attributes:

The INVIQ Inventory Management System prioritizes the following quality attributes to ensure a high level of performance, reliability, and usability for both customers and developers:

1. Usability: The system should be intuitive and user-friendly, with a clean and organized interface that allows users to navigate seamlessly and perform tasks efficiently. Usability testing will be conducted to ensure that the system meets usability standards, with a target usability score of at least 80% on standardized usability metrics.

2. Reliability: The system should operate consistently and reliably under normal and peak load conditions, with minimal downtime and errors. The Mean Time Between Failures (MTBF) should exceed 1000 hours, and the Mean Time to Repair (MTTR) should be less than 1 hour to ensure prompt resolution of any issues.

3. Maintainability: The system should be designed with modular architecture, clear code structure, and comprehensive documentation to facilitate ease of maintenance and future enhancements. Code maintainability metrics, such as code churn and cyclomatic complexity, should be tracked and optimized for continuous improvement.

4. Scalability: The system should be scalable to accommodate growing data volumes, user traffic, and business expansion. Scalability tests will be conducted to ensure that the system can handle a 50% increase in concurrent users and data load without degradation in performance.

5. Security: The system should enforce robust security measures to protect sensitive data, prevent unauthorized access, and mitigate security threats. Security assessments, penetration testing, and compliance audits will be conducted regularly to ensure adherence to industry standards and regulatory requirements.

6. Interoperability: The system should be compatible with existing hardware, software, andexternal services to facilitate seamless integration and interoperability. Standardized data formats, APIs, and communication protocols will be implemented to enable interoperability with third-party systems and services.

7. Testability: The system should be designed with testability in mind, with clear requirements, well-defined test cases, and automated testing frameworks in place. Test coverage metrics should be tracked, with a target test coverage of at least 80% for critical system components.

8. *Adaptability: The system should be adaptable to changing business requirements, technological advancements, and market dynamics. Agile development methodologies will be employed to facilitate iterative development, feedback-driven improvements, and rapid adaptation to evolving needs.*

Appendix A: Glossary

Glossary:

1. INVIQ: Acronym for Inventory IQ, the name of the inventory management system.

2. SRS: Software Requirement Specification, a document that describes the features, functionalities, and constraints of a software system.

3. GUI: Graphical User Interface, a visual way of interacting with a computer program through graphical icons and visual indicators.

4. DBMS: Database Management System, software that manages databases, enabling users to store, retrieve, and manipulate data.

5. API: Application Programming Interface, a set of rules and protocols that allows different software applications to communicate with each other.

6. MTBF: Mean Time Between Failures, a measure of the average time elapsed between system failures.

7. MTTR: Mean Time to Repair, a measure of the average time taken to repair a system after a failure occurs.

8. GDPR: General Data Protection Regulation, a set of data protection regulations in the European Union designed to protect the privacy and personal data of individuals.

9. HIPAA: Health Insurance Portability and Accountability Act, a US law that regulates the handling of sensitive patient health information.

10. API: Application Programming Interface, a set of routines, protocols, and tools for building software applications.

11. UI: User Interface, the means by which users interact with a computer program.

12. QA: Quality Assurance, a process of ensuring that software products meet specified requirements and standards.

13. Scrum: An agile framework for managing software development projects.

14. Agile: A set of principles for software development based on iterative and incremental development.

15. Usability Testing: A method of evaluating a product by testing it with representative users to identify usability issues.

16. Penetration Testing: A security testing technique used to identify vulnerabilities in a system by simulating real-world attacks.

17. Compliance Audit: An assessment of whether a system or organization complies with relevant laws, regulations, and standards.

18. Interoperability: The ability of different systems or software to communicate and exchange data with each other.

19. Modular Architecture: An architectural approach that divides a system into smaller, independent modules or components

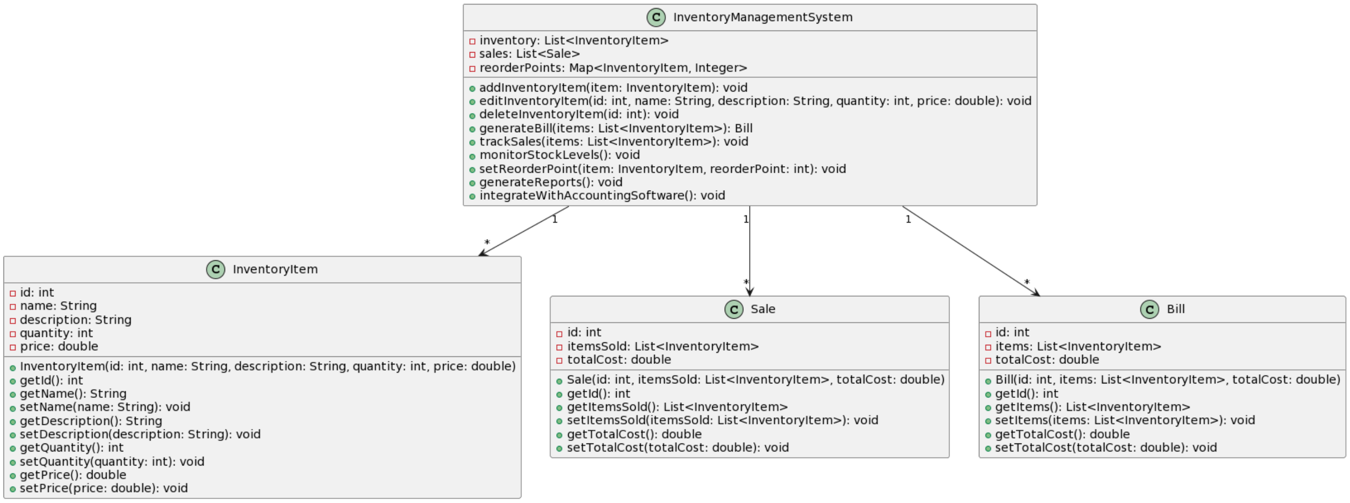
20. Test Coverage: A measure of the extent to which the source code of a program has been tested.

Appendix B: Analysis Models

Analysis Models:

1. Use Case Diagram:

- A visual representation of the system's functionality from the perspective of its users. It depicts various use cases and their relationships, illustrating how users interact with the system to accomplish specific tasks.


C InventoryManagementSystem
inventory: List<InventoryItem>
sales: List<Sale>
reorderPoints: Map<InventoryItem, Integer>
addInventoryItem(item: InventoryItem): void
editInventoryItem(id: int, name: String, description: String, quantity: int, price: double): void
deleteInventoryItem(id: int): void
generateBill(items: List<InventoryItem>): Bill
trackSales(items: List<InventoryItem>): void
monitorStockLevels(): void
setReorderPoint(item: InventoryItem, reorderPoint: int): void
generateReports(): void
integrateWithAccountingSoftware(): void
1
*
1
*
1
*
C InventoryItem
id: int
name: String
description: String
quantity: int
price: double
InventoryItem(id: int, name: String, description: String, quantity: int, price: double)
getId(): int
getName(): String
setName(name: String): void
getDescription(): String
setDescription(description: String): void
getQuantity(): int
setQuantity(quantity: int): void
getPrice(): double
setPrice(price: double): void
C Sale
id: int
itemsSold: List<InventoryItem>
totalCost: double
Sale(id: int, itemsSold: List<InventoryItem>, totalCost: double)
getId(): int
getItemsSold(): List<InventoryItem>
setItemsSold(itemsSold: List<InventoryItem>): void
getTotalCost(): double
setTotalCost(totalCost: double): void
C Bill
id: int
items: List<InventoryItem>
totalCost: double
Bill(id: int, items: List<InventoryItem>, totalCost: double)
getId(): int
getItems(): List<InventoryItem>
setItems(items: List<InventoryItem>): void
getTotalCost(): double
setTotalCost(totalCost: double): void

2. Entity-Relationship Diagram (ERD):

- A graphical representation of the data model that defines the relationships between entities in a system. It illustrates how different entities are related to each other through their attributes and associations.

3. Data Flow Diagram (DFD):

- A diagram that represents the flow of data within the system. It shows how data moves between different processes, data stores, and external entities, helping to visualize the system's data processing and information flow.

4. Class Diagram:

- A diagram that represents the static structure of the system by depicting the classes, attributes, methods, and relationships between objects. It provides a high-level overview of the system's object-oriented design and helps in understanding the system's architecture.

These analysis models aid in understanding the system's requirements, behavior, and structure, facilitating effective communication between stakeholders and guiding the design and development process. They help in identifying system components, defining system behavior, and establishing relationships between various elements of the system architecture.